Creativity, Design and Computer

Ewa Grabska Institute of Computer Science, Jagiellonian University
Nawojki 11, 30-072 Krakow, Poland
uigrabsk@cyf-kr.edu.pl

Abstract

This paper deals with computational aspects of creative design. A new approach to structural design representations in the framework of graph transformations is discussed. The representations are generated and modified by graph operations. Graphs represent structures of designs. A new software tool which enables designers to transform the design specification into the graph design structure is proposed. Examples of designing gardens and floor layouts for one-storey houses illustrate the presented methodology.

Keywords: CAD, creative design, hierarchical graph, graph operations, composite representation.

1 Introduction

This paper is concerned with computational aspects of creative design.
The development of creative design methods aided by computer is still in its infancy. There are reasons for this situation. First of all design eludes formal description. Furthermore the concept of creativity can mean very different things.
When tackling a design task for the first time designers try to handle the task by free-hand sketches. They, at the early stage, create rough drawings which help them to induce images of the entity that is being designed. CAD tools still appear to be too cumbersome for genuine sketching.
This paper does not consider the next computer imitation of traditional tools of sketching but tries to show that a chance of succeeding in using computers in the process of creative design is elsewhere. Such a chance lies in persuading the designer to use CAD tools which force him/her to think about the object that is being designed at 2 levels: the higher level of structural properties (syntax) and the lower level of geometry, primitives, attributes etc. (semantics). It is so because the real benefit of CAD is defining both syntax and semantics of the designed objects in an explicit way.
The proposed methodology used in design is based on a composite representation (Grabska and Borkowski, 1996) which assumes that the object graph structure is seperated from its realisation. The composite representation of a design consists of a graph reflecting the object structure and a realisation scheme for interpreting the graph in the form of a graphical representation of an artefact. Several types of graphical representations from line drawings to rendered images can be used for

presenting artefacts.
The main goal of this paper is to present graph operations as a way of handling a design task. They allow to consider aspects of creative design on the higher level of abstraction and provide a new method of creating non-recursive structures of designed objects. They are not only used for generation of structural representations but also enable modifications of object representations and facilitate to explore a space of potential design solutions by supporting reasoning about design on the syntactic level.
From the aesthetic point of view, the operations suggest syntactic characteristics of beauty. Such characteristics can be insufficient but can help the designer to be aware of ideas important for achieving desirable aesthetics of the object that is being created. In other words, application of such operations for describing object structures gives a chance that the probability of the objects attaining an acceptable appearance will enhance.
If design process begins with specifying the purpose then the reason for which an artefact is created is mapped into functions which enable to achieve it. The designer considers the decomposition of an artefact and the correspondence between the smaller units and the functions. The structure of the object that is being designed emerges from these considerations (Borkowski and Grabska, 1998).
The next goal of this paper is to present the Functional Structure Edytor (FSE), i.e., a new software tool which enables designers to transform the design specification into the composite representation and then to modify this structure by means of graph operations. FSE is a prototype program written in Visual Basic developed by the author and her students.

2 Composition Graphs and Interpretation

Our approach to design is *atomistic*. Artifacts are formed from simple buiding bloks called *primitives*. Given the primitives, we define the rules limiting the way they may be combined together. These rules determine the *syntax* of an artifact and allow us to define its structure. The structure of an artifact is described by a *composition graph* (*CP-graph*) (Grabska, 1993). In the definition of CP-graphs, the pattern theory developed by Grenander was used as a helpful framework (Grenander, 1972). Grenander drew his inspiration from Wittgenstein's Tractatus Logico-Philosophicus (Wittgenstein, 1955).
CP-graphs have nodes equipped with *bonds*. Given a node, in-bonds and out-bonds are distinguished (see: Fig. 1a). We draw the picture of a CP-graph representing bonds as small circles placed in the nodes and showing edges as the lines connecting the pairs of bonds (see: Fig. 1b). The number of node bonds expresses the maximal number of the connections between the primitive corresponding to the node and other primitives. A CP-graph has two kinds of bonds: *engaged* bonds which are connected by means of edges and the remaining bonds called *free* bonds. Each node

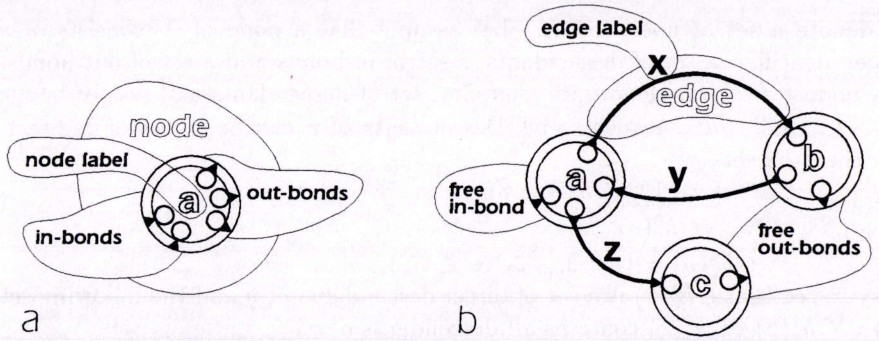

Fig. 1: The CP-graph a) its node b) its structure.

has a label. When mapping a CP-graph into a graphical representation the label determines a primitive (a basic geometrical shape) whose transformed copy (a graphical component) is assigned to the node. Edges indicate the existence of different relations between graphical components. Names of these relations constitute edge labels.

Graph operations require an extension of CP-graphs to hierarchical graphs. We assume that nodes can contain inner nodes called their *descendants.* A descendant can also have bonds and be connected to other nodes. Although there exists an exception - any node can not be connected by edges with its ancestors. Descendants can be visible or hidden, depending on details required by the designer. Fig. 2a shows an example of the CP-graph with its visible descendants. Some of them are hidden in Fig. 2b.

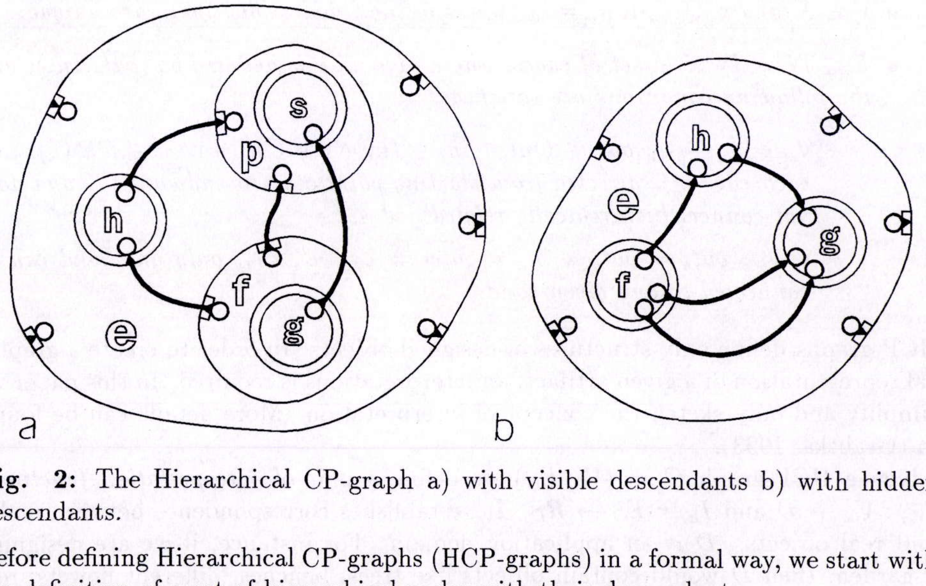

Fig. 2: The Hierarchical CP-graph a) with visible descendants b) with hidden descendants.

Before defining Hierarchical CP-graphs (HCP-graphs) in a formal way, we start with the definition of a set of hierarchical nodes with their descendants.

Let us denote a set of nodes by X. Let assume that a node of X consists of a numerical identifier, a set of descendants, a set of in-bonds and a set of out-bonds. Given a node $v \in X$, we denote its identifier, set of descendants and sets of bonds by i_v, C_v, In_v and Out_v, respectively. Descendants of v can be direct or indirect. Let us define the set:

$$Ch^*(v) = \bigcup_{n=0}^{\infty} Ch^n(v), \text{ where}$$
$$\forall_{v \in X} Ch^0(v) = v$$
$$Ch^n(v) = \{w : \exists_{u \in Ch^{n-1}} w \in C_u\}$$

It is easy to see that $Ch^1(v)$ is a set of direct descendants of v and the following set $Ch^+(v) = Ch^*(v) \setminus Ch^0(v)$ contains all descendants of v.

Definition 1 *By a set of hierarchical nodes we understand a set*

$$X = \{v : v \in \bigcup_{n=0}^{\infty} X_n, \forall_{w \in Ch^*(v)} (In_w \cup Out_w) < \infty\}, \textit{where}$$

$$X_0 = \{(i, C, In, Out) : i \in I\!N, C = \emptyset, In \subset I\!N \times \{i\}, Out \subset \{i\} \times I\!N\}$$

$$X_n = \{(i, C, I_n, Out) : i \in I\!N, C \subset \bigcup_{k=0}^{n-1} X_k, In \subset I\!N \times \{i\}, Out \subset \{i\} \times I\!N\}, n \geq 1$$

Definition 2 *Let X be a set of hierarchical nodes and $Ch^*(v)$ be a set consisting of a given node $v \in X$ and all its descendants. By a Hierarchical CP-graph (HCP-graph) we mean a pair (V, E), where*

- *$V \in X$ and $\forall_{v,w \in Ch^*(V)} i_v = i_w \Rightarrow v = w$, i.e., nodes identifiers are unique,*
- *$E \subset I\!N^2 \times I\!N^2$ is a set of edges, where edge an e is denoted by (out_e, in_e), and the following conditions are satisfied:*
  - *$\forall_{e \in E} \exists_{v,w \in Ch^*(V)} out_e \in Out_v \wedge in_e \in In_w \wedge v \notin Ch^*(w) \wedge w \notin Ch^*(v)$, i.e., each edge e is directed from starting point out_e to endpoint in_e and does not connect hierarchically related nodes,*
  - *$\forall_{e,e' \in E} (out_e = out_{e'} \vee in_e = in_{e'}) \Rightarrow e = e'$, i.e., only one bond can be connected to any given bond.*

HCP-graphs define only structures of designed objects. In order to create a graphical representation of a given artifact, an interpretations is required. In this paper we simplify and only sketch the concept of interpretation. More details can be found in (Grabska, 1993).

Given a HCP-graph $G = (V_G, E_G)$ we define a pair of *interpretation functions*: $I_{V_G} : V_G \rightarrow D$ and $I_{E_G} : E_G \rightarrow R_D$. I_{V_G} establishes correspondence between nodes and real objects. D is an application domain. For instance, if we are designing a garden, then D would contain objects like trees, benches, different flowers, recreation grounds, lawns, etc. I_{E_G} assigns relations defined between objects of D to edges. R_D is a set of possible binary relations defined on D, for instance "to the north", "bigger", etc.

3 Graph Operations

There are many reasons why graph operations should be introduced. First, structural properties which can be handled by means of them are essential in inducing images of the entity that is being designed. Secondly, evaluating potential design solutions often suggests the designer some improvements and modifications. As a consequence, structural changes can be necessary to obtain better results. Moreover, graph operations allow to describe general aesthetic characteristics common to attractive objects on the syntactic level. To show how the proposed graph operations handle design tasks we shall present designing gardens.
Four important operations on CP-graphs are:

- gluing operation,
- merging operation,
- splitting operation, and
- concatenation.

3.1 Gluing operation

Designing is contextual. Gluing operation reflects this fact. Creating components of the designed object should include their context. Let us consider a class of gardens which contain both a patio and an extra recreation ground. We assume that adding next components of gardens is contextual, i.e., the components are placed between a patio and a recreation ground. Besides the context our preliminary design contains a rock garden and a lawn. Adding a flower bed to the central part of the designed garden requires a gluing operation (see: Fig. 3a). We glue the context and add the next component to the central part.
A gluing operation of two drawings shown in Fig. 3a requires the existence of the common, i.e. (context) in these drawings. The two elements (recreation ground and patio) belonging to both drawings satisfy this requirement. In this example the gluing operation can be seen as adding a new fragment of the second drawing to the first drawing with preserving their context.
Let us consider this example using the gluing operation for CP-graphs which represent the drawing structures. The CP-graphs depicted in Fig. 3b correspond to the drawings shown in Fig. 3a.
The CP-graph having 4 nodes and 4 edges is glued with the CP-graph with 3 nodes and 2 edges. The edges are determined by the relation *being below*. The result of gluing operation is the CP-graph which has 6 edges and 5 nodes. Its nodes with

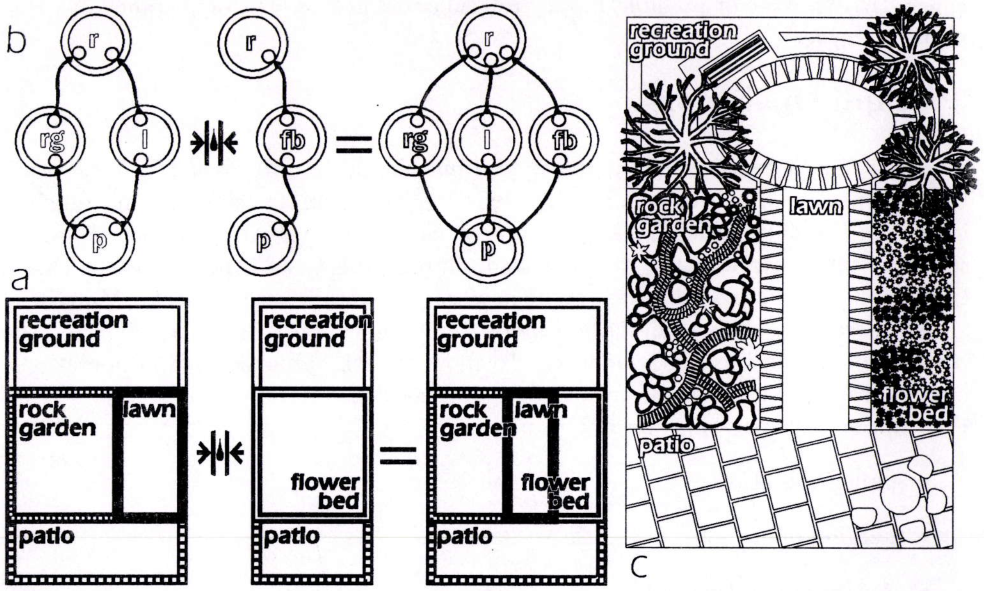

Fig. 3: Gluing operation: a) user interface; b) CP-graph representation; c) interpretation of the result.

labels r and p are the results of gluing two pairs of nodes of the glued CP-graphs, which have label r and p, respectively. The edges of glued nodes are added. Therefore in the resulted CP-graph the number of in-bonds of the node with label r and the number of out-bonds of the nodes with labels p are equal to 3.
In Fig. 3c we can see one of possible interpretations of the resulted CP-graph.
In general, a gluing operation can be seen as an operation of matching subgraphs. Matching, described by means of two functions, consists in detecting subgraphs which are "similar enough" and can be candidates for gluing. The matching functions work in a hierarchical way - two nodes can be considered to be equal only if their parents are equal. These functions should base its decisions not only on the graph structures alone, but also on details found in the interpretation. Two nodes assumed to be equal do not have to have identical interpretations. When two nodes with different interpretations are glued into one? An answer to this question has to provided by the matching functions.
Before the formal definition of matching let us define a set $B(V)$ of all bonds for a given HCP-graph $G = (V, E)$:

$$B(V) = \{b \in \mathbb{N}^2 : \exists_{w \in Ch^*(V)} b \in In_w \cup Out_w\}$$

As it has been considered bonds are attached to the nodes. Let us denote the node v with bond b by $v(b)$.

Definition 3 *Let $G = (V, E)$ and $G' = (V', E')$ be two HCP-graphs. By a matching M for G and G' we mean a pair of functions (mv, mb), where*
$mv : Ch^*(V) \times Ch^*(V') \to \{0, 1\}$ and $mb : E \times E' \to \{0, 1\}$,
satisfying the following conditions:

- $mv(v, w) = 1 \Rightarrow (\forall_{u \in V, u \neq v} mv(u, w) = 0 \wedge \forall_{u' \in V', u' \neq w} mv(v, u') = 0)$, *i.e., for any given node, there is no more than one matching,*
- $mv(v, w) = 1 \Rightarrow (\forall_{u \in Ch^*(V)} \forall_{u' \in Ch^*(V')} v \in Ch(u) \wedge w \in Ch(u') \Rightarrow$ $\Rightarrow mv(u, u') = 1)$, *i.e., if two nodes match, their parents also must match or these nodes are topmost nodes in their HCP-graphs,*
- $mb(a, b) = 1 \Rightarrow (\forall_{c \in B(V), c \neq a} mb(c, b) = 0 \wedge \forall_{d \in B(V'), d \neq b} mb(a, d) = 0)$, *i.e., for any given bond, there is no more than one matching,*
- $mb(a, b) = 1 \Rightarrow mv(v(a)) = mv(v(b)) = 1$, *i.e., if two bonds match, their nodes also must match,*
- $mb(a, b) = 1 \Rightarrow (\exists_{v \in Ch^*(V)} a \in Out_v \leftrightarrow \exists_{v' \in Ch^*(V')} b \in Out'_v$, *i.e., out-bonds can match only other outbonds, in-bonds only in-bonds.*

Before our consideration of gluing operation let us define a set $I(V)$ of all identifiers for V:

$$I(V) = \{i \in \mathbb{N} : \exists_{w \in Ch^*(V)} i = i_w\}$$

Let $G = (V, E)$ and $G' = (V', E')$ be two HCP-graphs for gluing and let $H = (V_H, E_H)$ be the result of this operation. G and G' must have unique identifiers: $I(V) \cap I(V') = \emptyset$.
If topmost nodes do not match, then nothing else will match, and both HCP-graphs will be copied fully. We have to create a new node and link V and V' as its children, because there can be only one topmost node in each HCP-graph.
On the other hand, if V and V' match, then they will be glued, and we have to test their children to see whether some of them can be glued also.

$$V_H = \begin{cases} (i, \{V, V'\}, \emptyset, \emptyset), i \notin I(V) \cup I(V') : & \text{mv(V,V')} = 0 \\ U(V, V') : & \text{mv(V,V')} = 1 \end{cases}$$

where $U(V, V') = (i_V, UC(C_V, C_{V'}), UB(In_V, In_{V'}), UB(Out_V, Out_{V'}))$ is a result of gluing nodes, where
$UC(C_V, C_{V'}) = \{U(y, z) : \exists_{y \in C_V} \exists_{z \in C_{V'}} mv(y, z) = 1\} \cup$
$\cup \{y : y \in C_V \neg \exists_{z \in C_{V'}} mv(y, z) = 1\} \cup \{z : z \in C_{V'} \neg \exists_{y \in C_V} mv(y, z) = 1\}$ is a result of gluing descendants, and
$UB(A, B) = \{a : \exists_{a \in A} \exists_{b \in B} mb(a, b) = 1\} \cup \{a : a \in A \neg \exists_{b \in B} mb(a, b) = 1\} \cup$
$\cup \{b : b \in B \neg \exists_{a \in A} mb(a, b) = 1\}$ is a result of gluing bonds.
After defining the set V_H of nodes the set E_H of edges is determined in the following way: if two edges have matching bonds on one end, then bonds on the other end

must mutch also, and we glue these edges. Otherwise, we just copy them. In a formal way:
$E_H = \{(a,b) : (a,b) \in E_V \exists_{(c,d) \in E_{V'}} mb(a,c) = 1\} \cup$
$\cup \{(a,b) : (a,b) \in E_V \neg\exists_{(c,d) \in E_{V'}} mb(a,c) = 1\} \cup$
$\cup \{(c,d) : (c,d) \in E_{V'} \neg\exists_{(a,b) \in E_V} mb(a,c) = 1\}.$
Interpretations of newly created graph H is based on interpretations of V and V'.

3.2 Merging operation

A certain amount of hierarchy is necessary when keeping in mind rationality and efficiency of the design process. The merging operation allows one to group together

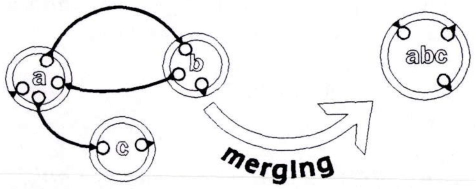

Fig. 4: Merging the subgraph with three nodes into one node.

nodes of a CP-graph and to represent them by one hypernode. This operation ignores the structure of nodes which are grouped but preserves their free-bonds which

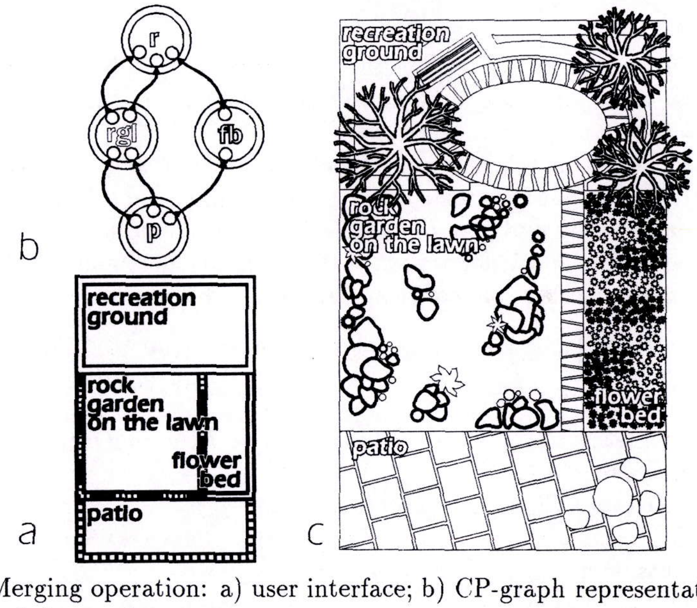

Fig. 5: Merging operation: a) user interface; b) CP-graph representation; c) interpretation of the result.

represent connections of the formed hypernode with its environment. In other words, a node which is substituted for a subgraph formed by nodes which are grouped has

the same number of in-bonds and out-bonds as the number of free in-bonds and out-bonds, respectively, of the replaced subgraph (see: Fig. 4).
Let us consider the resulted CP-graph in Fig. 3b. The merging operation is applied to two nodes with labels *rg* and *l* of this CP-graph. These nodes are replaced by one node (hypernode) with label *rgl*. The result of the merging operation is shown in Fig. 5b. Fig. 5a presents the drawing corresponding to the resulted CP-graph. Components of the drawing being the rock garden and the lawn are joined. The interpretation of this CP-graph is the rock garden on the lawn (see: Fig. 5c).
In general, merging operation can be applied to HCP-graphs. This operation defined for a given HCP-graph consists in removing all descendants from the graph. The result graph is the topmost node.

3.3 Splitting operation

Apart from the merging operation, also an inverse operation proves to be helpful in design. The splitting operation adds an internal structure. It represents division of

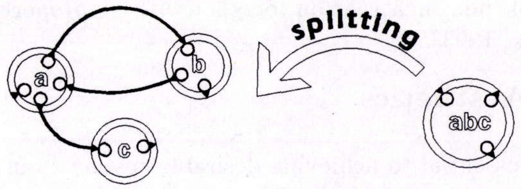

Fig. 6: Splitting the node into the subgraph with three nodes.

components into smaller parts by replacing one node with a CP-graph having the same number of free in-bonds and free out-bonds as the number of in-bonds and out-bonds, respectively, of the replaced node (see: Fig. 6). The splitting operation can be applied to the node representing the recreation ground. This node is divided

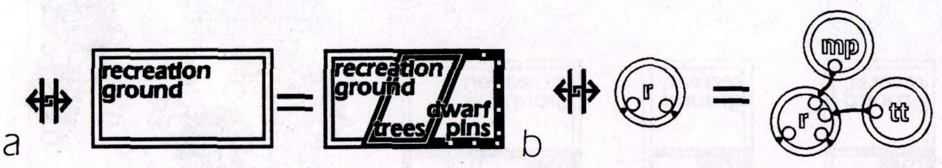

Fig. 7: Splitting operation: a) user interface; b) CP-graph representation.

into three nodes. Two nodes which represent trees and dwarf pine are seperated from the recreation ground. Fig. 7 presents this operation.
In general, the splitting operation can be applied to HCP-graphs. This operation defined for a given HCP-graph consists in adding some descendants to this graph.

3.4 Concatenation

The concatenation operation does not require any context in a design. Generally, concatenating CP-graphs consists in creating edges between free bonds of these CP-graphs.

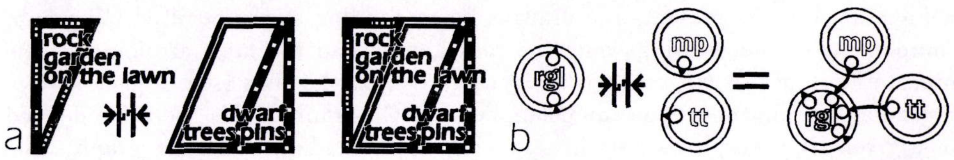

Fig. 8: Concatenation operation: a) user interface; b) CP-graph representation.

Let us consider the result of the splitting operation shown in Fig. 7. The two nodes representing trees an dwarf pine are connected with the node corresponding to the rock garden on the lawn (see: Fig. 8). In the following the CP-graph being the result of the concatenation operation will be essential in the structure describing a regular garden.

A formal definition of the concatenation together with its properties is presented in (Grabska and Hliniak, 1993).

4 The Key to Aesthetics

In this section ideas essential to achieving desirable aesthetics in object design are

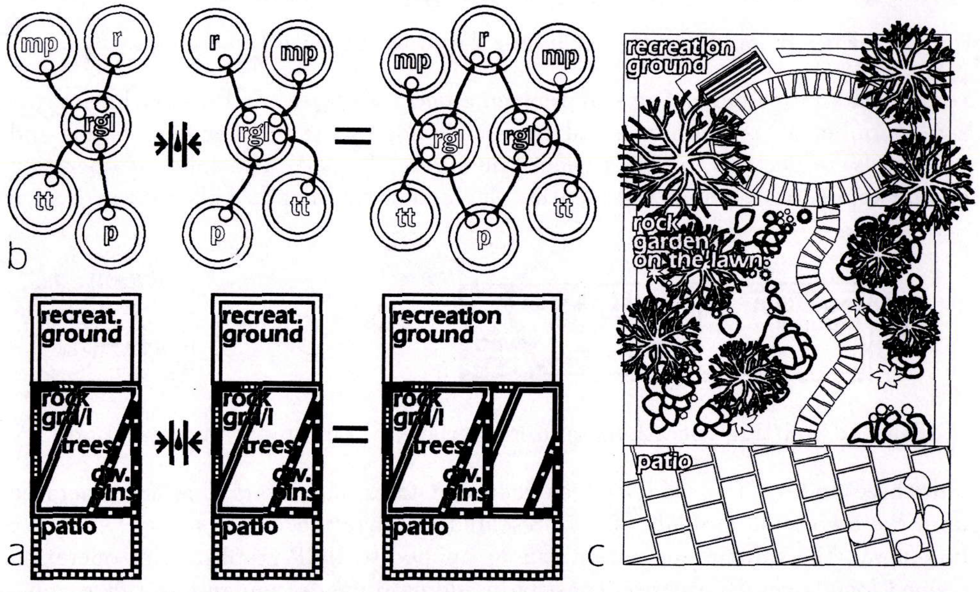

Fig. 9: Visual balance: a) user interface; b) CP-graph representation; c) interpretation of the result.

sketched. An internal structure of the object has a great influence on its form. This idea is used as a basis for the esthetic considerations presented here.
It is difficult to define beauty. Through the study of aesthetics it is possible to i-dentify general characteristics common to attractive objects. Although possesion of such characteristics does not guarantee beauty, they greatly enhance the probability of the object attaining an acceptable appearance.
Characteristics for beauty in object design are identified as unity and harmony (Tjal-ve, 1978). They can be achieved through balance, rhythm and proportion. Syntactic level restricts the discussion to balance and rhythm.
Usually, in both symmetric and asymmetric designs, visual balance is seen, as exis-tence of the same visual weight on both sides of the object's centre line.
A structure of symmetric design with visual balance is considered as the result of gluing operation of two copies of the same CP-graph which is a proper subgraph of the total CP-graph design structure (see: Fig. 9a). The garden being a symmetric

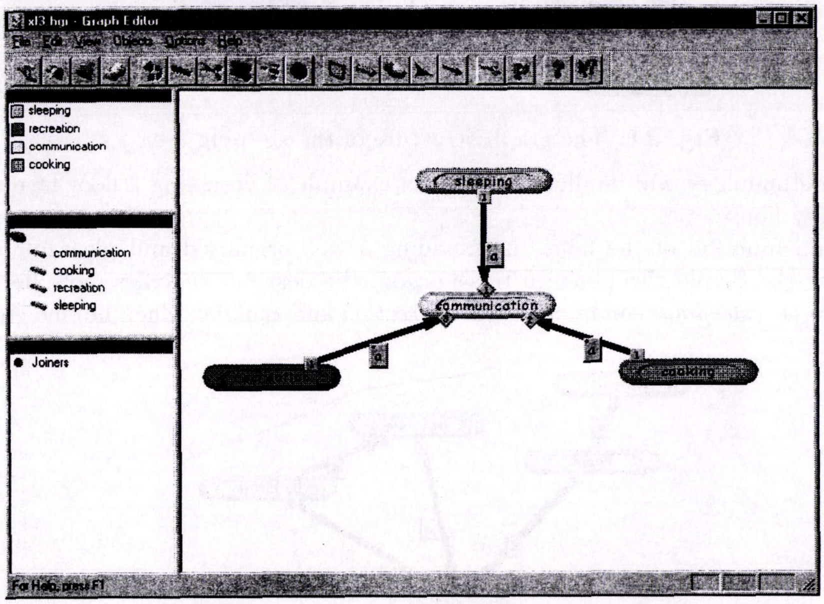

Fig. 10: Converting functions into the graph.

design with visual balance is depicted in Fig. 9c.
Rhythm refers to repetition of patterns and can be achieved through variations of ar-rangement, colour, etc. Operation of concatenation allows to connect such fragments which preserve appropriate arrangements. By grouping elements into hypernodes the hypernode operation makes it possible to preserve rhythm. For instance, this may be achieved by applying the same colour to mark the distinguished hypernodes.

5 Converting Functions into Object Structure

Usually, design process begins with specifying the purpose, i.e., the reason for which an artefact is created. In this section we consider defining functional requirements of the designed object and transforming them into the object strucure. The pro-

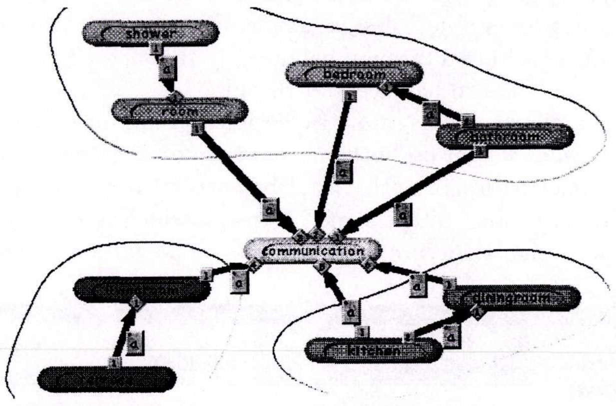

Fig. 11: The graph structure of the sleeping area.

posed methodology will be illustrated by an example of designing a floor layout for one-storey house.

The main function of the house is providing a well organised and pleasant living space for the family. To perform this purpose the designer specifies more detailed functions as: *sleeping, communication, recreation* and *cooking.* Then he/she creates

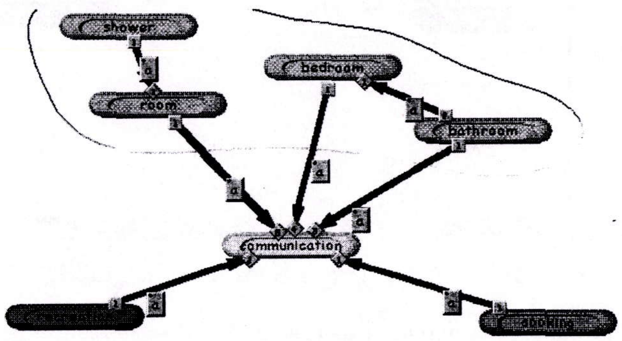

Fig. 12: The preliminary layout

a graph composed of four nodes, where each of them represents one of these functions. To convert these functions into the graph the designer can use the Function Structure Editor (FSE) (see: Fig. 10). FSE allows the designer to apply graph operations. For instance, when applying splitting operations each node can be replaced by several different subgraphs as one can have different arrangements of the rooms playing the same roles. One of the possible structures of the sleeping area is shown in Fig. 11. The node *sleeping* is replaced by a subgraph composed of four nodes,

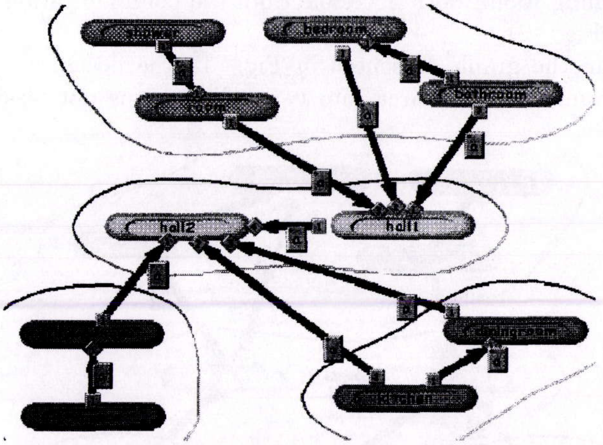

Fig. 13: The first way of establishing the accessibility relations.

where three of them, representing a bedroom, room and bathroom, are connected with the node *communication.* These connections represent the accessibility of the three places from a hall which will be placed inside the house. Moreover, the node *shower* is connected with the node *room* and the node *bathroom* is connected with

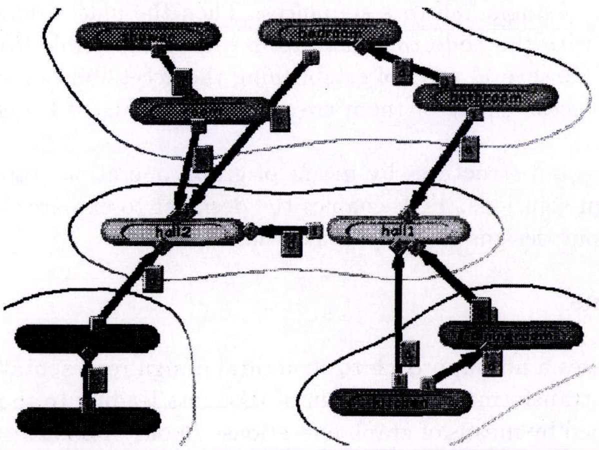

Fig. 14: The second way of establishing the accessibility relations.

the node *bedroom* as the shower and the bathroom should be accesible from the appropriate places.
The node *recreation* is replaced using a splitting operation by a two-node graph,

where the first node represents a living room accessible from the communication area, while the second one corresponds to a terrace accessible from the lining room. The node *cooking* is replaced also by a two-node graph whose nodes represent a kitchen and a dining room, both accesible from the communication area and from each other (see: Fig. 12).
While considering the graph presented in Fig. 12 the designer can suggest the division of the communication area into two halls as it is not feasible to arrange

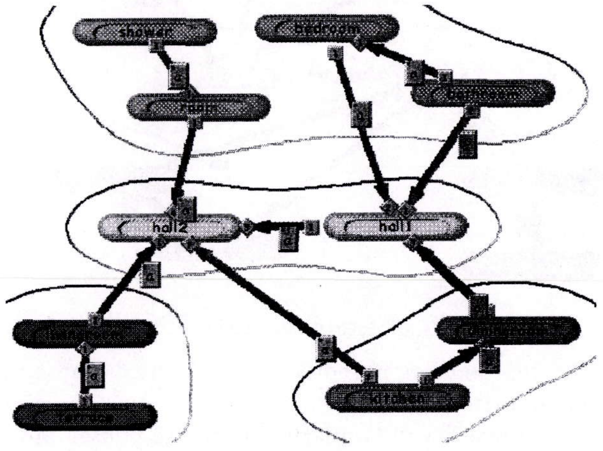

Fig. 15: The third way of establishing the accessibility relations.

direct access from a single hall to a six places. Then the places represented by the nodes connected with the node *communication* will be accessible from one of these two halls. There are several ways of establishing the accessibility relations between halls and other places. Three of them are presented in Fig. 13, Fig. 14, and Fig. 15.
Modifications of graph structures by means of graph operations constitute a space of potential design solutions. FCE enables the designer to explore the design space and to reason about designs on the syntactic level.

6 Conclusions

This paper discusses a new approach to structural design representations in the framework of graph transformations. A train of thoughts leading to the creation of an artefact is described by means of graph operations. In our opinion advanced human-computer interface with successful implementation of graph operations allows the computer to provide information to the designer, related to structural properties of the entity that is being designed.
Moreover, our attention has been focused on a tool enable us to bridge a gap between specification and the object structure being an element of a composite representa-

tion. It would be naive to imagine that the specification of functional requirements of the designed object could be automatically translated into its structure. This mapping is obviously non-unique and many crucial decision should be taken by the designer. But the aim of FSE is to assist him/her in the descision-making process and to facilitate achieving the plausible design solutions.

References

Borkowski Adam and Grabska Ewa (1998); Converting Function into Object; in Smith, I. (ed); Lecture Notes in Artificial Intelligence, 1454, pp. 434-439; Springer-Verlag, Berlin.

Grabska Ewa (1993); Theoretical Concepts of Graphical Modeling: Realization of CP-graphs; Machine GRAPHICS and VISION, 2(1), pp.3-38, Warszawa.

Grabska Ewa and Borkowski Adam (1996); Assisiting Creativity by Composite Representation; in Gero, J.S. and Sudweeks, F. (eds); Artificial Intelligence in Design'96; pp.743-749; Kluwer Academic Publishers.

Grabska Ewa nad Grazyna Hliniak (1993): Structural aspects of CP-graph Languages, Zeszyty Naukowe UJ, Prace Informatyczne, 5.

Grenander Ulf (1976); Pattern Synthesis, Springer Verlag.

Tjalve Eskild (1978); Systematische Formgebung fur Industrieprodukte, VDI-Verlag GmbH, Dusseldorf, und FACHPRESSE GOLDACH, Hudson&Co.

Wittgenstein Ludwig (1955); Tractatus Logico-Philosophicus; Sixth Edition, London.